Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

IIC2343 – Arquitectura de Computadores

Ayudantía 2

Profesor: Yadran Eterovic

Ayudantes: Daniel Leal (dlleal@uc.cl), Jessica Hormazabal (jyhormazabal@uc.cl)

Preguntas

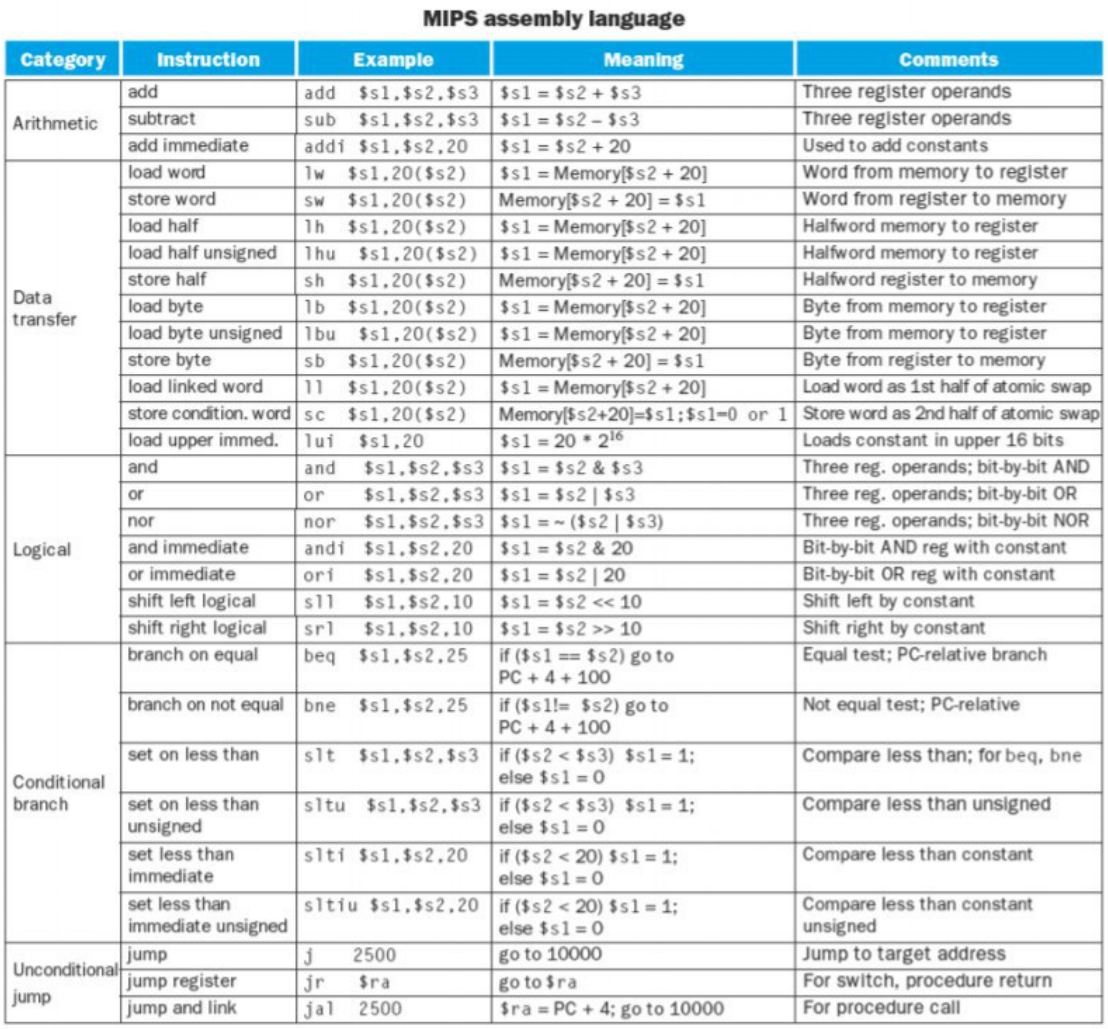
1. **Assembly**

   a) Cargue los valores 5 y 17 en los registros $s1 y $s2. Considere registros de 1 byte.

   b) Utilizando la arquitectura ISA MEEP MEEP, programe una multiplicación en *assembly* entre dos números guardados en memoria. Considere que los números a multiplicar están en las direcciones 8 y 9, mientras que el resultado desea guardarse en la dirección 13.

   c) Utilizando la arquitectura ISA MEEP MEEP, programe en *assembly* un código que le permita sumar N números desde el número P_0 en adelante. Asuma que N y P_0 están guardados en la memoria RAM en las direcciones 13 y 21, respectivamente. Guarde el valor final en la dirección 4 de la memoria. Comente si lo encuentra necesario.

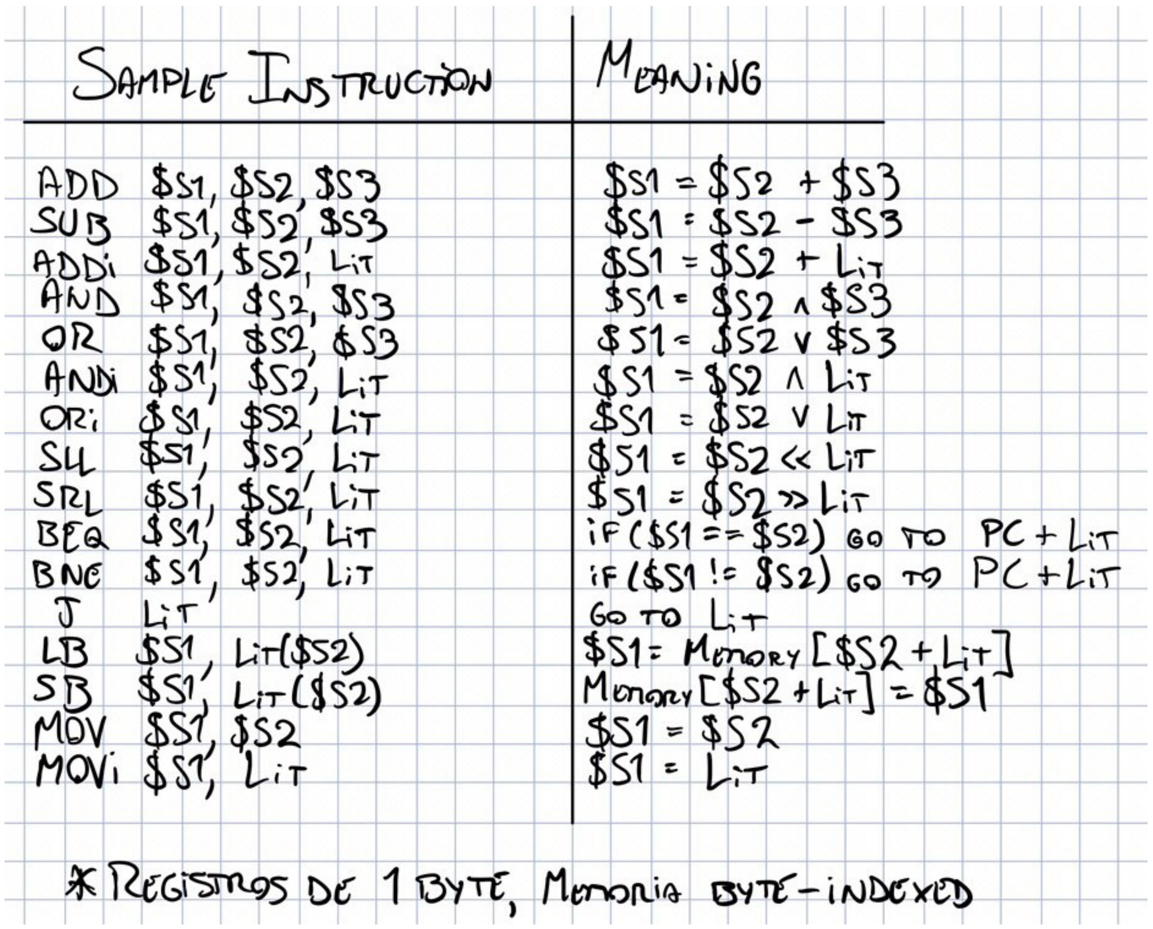
2. **Anexos**



MIPS assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add $s1,$s2,$s3	$s1 = $s2 + $s3	Three register operands
	subtract	sub $s1,$s2,$s3	$s1 = $s2 – $s3	Three register operands
	add immediate	addi $s1,$s2,20	$s1 = $s2 + 20	Used to add constants
Data transfer	load word	lw $s1,20($s2)	$s1 = Memory[$s2 + 20]	Word from memory to register
	store word	sw $s1,20($s2)	Memory[$s2 + 20] = $s1	Word from register to memory
	load half	lh $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	load half unsigned	lhu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	store half	sh $s1,20($s2)	Memory[$s2 + 20] = $s1	Halfword register to memory
	load byte	lb $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	load byte unsigned	lbu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	store byte	sb $s1,20($s2)	Memory[$s2 + 20] = $s1	Byte from register to memory
	load linked word	ll $s1,20($s2)	$s1 = Memory[$s2 + 20]	Load word as 1st half of atomic swap
	store condition. word	sc $s1,20($s2)	Memory[$s2+20]=$s1;$s1=0 or 1	Store word as 2nd half of atomic swap
	load upper immed.	lui $s1,20	$s1 = 20 * 2^{16}	Loads constant in upper 16 bits
Logical	and	and $s1,$s2,$s3	$s1 = $s2 & $s3	Three reg. operands; bit-by-bit AND
	or	or $s1,$s2,$s3	$s1 = $s2 | $s3	Three reg. operands; bit-by-bit OR
	nor	nor $s1,$s2,$s3	$s1 = ~ ($s2 | $s3)	Three reg. operands; bit-by-bit NOR
	and immediate	andi $s1,$s2,20	$s1 = $s2 & 20	Bit-by-bit AND reg with constant
	or immediate	ori $s1,$s2,20	$s1 = $s2 | 20	Bit-by-bit OR reg with constant
	shift left logical	sll $s1,$s2,10	$s1 = $s2 << 10	Shift left by constant
	shift right logical	srl $s1,$s2,10	$s1 = $s2 >> 10	Shift right by constant
Conditional branch	branch on equal	beq $s1,$s2,25	if ($s1 == $s2) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne $s1,$s2,25	if ($s1!= $s2) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than; for beq, bne
	set on less than unsigned	sltu $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than unsigned
	set less than immediate	slti $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant
	set less than immediate unsigned	sltiu $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant unsigned
Unconditional jump	jump	j 2500	go to 10000	Jump to target address
	jump register	jr $ra	go to $ra	For switch, procedure return
	jump and link	jal 2500	$ra = PC + 4; go to 10000	For procedure call

Figura 1: ISA de un computador MIPS



SAMPLE INSTRUCTION	MEANING
ADD $S1, $S2, $S3	$S1 = $S2 + $S3
SUB $S1, $S2, $S3	$S1 = $S2 - $S3
ADDi $S1, $S2, LiT	$S1 = $S2 + LiT
AND $S1, $S2, $S3	$S1 = $S2 ∧ $S3
OR $S1, $S2, $S3	$S1 = $S2 ∨ $S3
ANDi $S1, $S2, LiT	$S1 = $S2 ∧ LiT
ORi $S1, $S2, LiT	$S1 = $S2 ∨ LiT
SLL $S1, $S2, LiT	$S1 = $S2 << LiT
SRL $S1, $S2, LiT	$S1 = $S2 >> LiT
BEQ $S1, $S2, LiT	iF ($S1 == $S2) GO TO PC + LiT
BNE $S1, $S2, LiT	iF ($S1 != $S2) GO TO PC + LiT
J LiT	GO TO LiT
LB $S1, LiT($S2)	$S1 = MEMORY [$S2 + LiT]
SB $S1, LiT($S2)	MEMORY [$S2 + LiT] = $S1
MOV $S1, $S2	$S1 = $S2
MOVi $S1, LiT	$S1 = LiT

* REGISTROS DE 1 BYTE, MEMORIA BYTE-INDEXED

Figura 2: ISA de un computador MEEP MEEP